REPRESENTATION AND MATCHING OF KNOWLEDGE TO DESIGN DIGITAL SYSTEMS

J. U. Jones
Rockwell International
555 Discovery Drive
Huntsville, Alabama 35805

S. G. Shiva
Computer Science Department
University of Alabama in Huntsville
Huntsville, Alabama 35899

ABSTRACT

In this paper, we describe a knowledge-based expert system that provides an approach to solve a problem requiring an expert with considerable domain expertise and facts about available digital hardware building blocks. To design digital hardware systems from their high level VHDL (Very High Speed Integrated Circuit Hardware Description Language) representation to their finished form, a special data representation is required. This data representation as well as the functioning of the overall system is described.

1. INTRODUCTION

Automatic hardware synthesis is important, because the complexity of integrated circuits has became as high as millions of transistors per device. Yet, the turn around time available for each design has become shorter and shorter due to competition. Design aids such as logic minimization tools, wire routers, simulation tools, and hardware synthesis systems have been developed; but they do not approach the efficiency of manual hardware design. Overviews of hardware synthesis systems can be found in [2,6].

The Department of Defense initiated the Very High Speed Integrated Circuit (VHSIC) Program to aid in the production of military integrated circuits [7]. To create an integrated design tool taking the designer through all phases of development, testing and evaluation, the VHSIC Hardware Description Language (VHDL) was developed [8,9]. Research in the area of developing an expert system which would function as an integrated design tool has been done at the University of Alabama in Huntsville (UAH) by Green [3] and Klon [4]. Green designed the University of Alabama Hardware Expert Synthesis System (UHESS) which serves as a prototype design consultant in the selection of VHSIC chips. Klon investigated the issues concerning interfacing UHESS to VHDL. He extended Green's knowledge base representation data structure and named it a hologram, which was used to synthesize sample hardware designs. This paper extends Klon's hologram representation.

2. THE EXPERT SYSTEM

To design digital hardware, first a VHDL source code is developed and input into the expert system. The expert system translates this source code into a hologram representation. The design library contains holograms of modules that have been designed earlier. These holograms can be at different levels of sophistication (detail). They may contain only one gate, a combi-

national circuit, a complex circuit like a printer interface board, or a whole microcomputer.

Once a hologram representation is given, the inference engine can begin to search the library for similar ones. The outcome of this search could be: a matching hologram found, a similar hologram or holograms are found, or a similar hologram does not exist. In each case the actions to be taken to define subholograms are described later in this paper.

When hologram representations are defined for the subholograms, the procedure described above can be used recursively until, the top hologram description is decomposed into a tree with nodes of subholograms where the leaf node holograms can be found in the library. The phase just described is the requirement decomposition phase. The next phase is the synthesis. Here children nodes are synthesized to become the parent hologram. This process begins at the leaf nodes and propagates actual design constraints upward on the tree until the root node is reached.

The difficulty lies in the fact that design uses nonmonotonic reasoning. The correctness of design assumptions can be verified only by exploring all consequences of the assumptions. Wrong assumptions can be taken during the analysis as well as the synthesis phase going from lower level nodes toward higher level subholograms. The source of incorrect decisions can be pinpointed and another design assumption has to be taken. All consequences of the previous incorrect decisions have to be traced and replaced with the consequences of the new decision. Therefore, the synthesis process can be described as a constant back and forth motion between analysis and synthesis, but eventually will end up at the root node. At that point the finished circuit design is output and incorporated into the library. Figure 1 shows the digital circuit design process.

Another difficulty is that in digital circuit design there is no one best design; there may be several equivalent good designs. There is a question wheather alternative designs are better. Or if they are better, what is it that makes them better? There are several options a design could be optimized for. Number of components, cost, speed, size, heat generated, and long life are a few examples. To optimize a design to any of these requirements or any combination of them requires a different approach. This problem could be solved two ways: one is that the hardware synthesizer develops all alternatives with a nondeterministic design approach; the other is that the physical device constraints are taken into account early on in the design. It is easily seen that the nondeterministic design approach would take many iterations, and in most cases would be inefficient. Therefore, the purely top down design methodology can not be used, and a combination of top down and bottom up design methodology is needed. Some actual device constraints have to be taken into consideration at the design specification (VHDL code generation) phase. This implies that the way the specification is written in VHDL has a very large effect on the implementation.

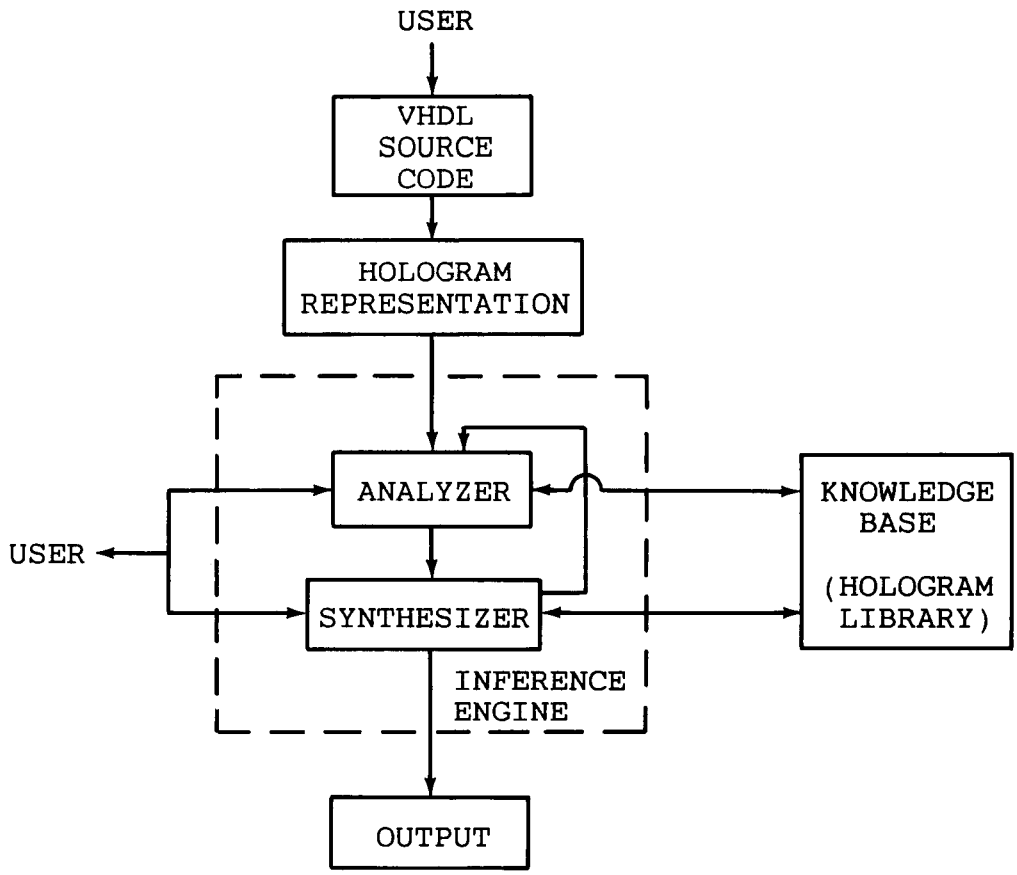

Figure 1. The digital circuit design process.

3. HOLOGRAM DATA STRUCTURE AS A DESIGN REPRESENTATION

Green's dissertation focused on the Knowledge Base (KB) representation for an Expert System which would design digital circuitry. It investigated issues such as knowledge acquisition tasks, the knowledge being represented, and proposed a knowledge representation scheme. This knowledge representation scheme combined frames and production rules to generate prototypes. This representation scheme was selected because it integrated the strengths of both representations. The prototype representation contains slots for knowledge storage. This knowledge consists of data pertaining to VHSIC chips and rules establishing default values and knowledge heuristics.

Klon investigated the issues of interfacing UHESS to VHDL and developed a scheme to represent dynamic semantics. The hologram which is a data structure to represent one or a collection of devices combines the features of prototypes and semantic networks. Modules (a collection of chips with their interfaces and functions) represent objects, functions, and activities, while signals are carriers of information between modules. Hierarchical semantic structures are propagated through modules; and semantics of the connectivity between modules are propagated through signals. Therefore, there is a need for two types of prototypes: module and signal. Holograms are a network of modules and sig-

nals. Figure 2 shows an abbreviated hologram for an ALU. The hologram description contains a hologram name, type, port assignments, element (or submodule description), a netlist which shows how the elements are interconnected, and rules containing information on the use of the hologram. Figure 3 depicts the abbreviated hologram structure for the ALU.

The hologram representation holds advantages over pure prototype or pure semantic network representation. Comparison of two

```
NAME:  ALU
TYPE:  MODULE
IN:  A, BUS
     B, BUS
     S, BIT
     CLK, BIT
     COMP, BIT
OUT:  C, BUS
LOCAL:  Y, BUS
ELEMENT ASSIGNMENTS:
      1, EIGHT_BIT_ADDER
      2, SHIFTER
      3, TWOS_COMP
NETLISTS:
      A; (1,1), (2,2)
      B; (3,2)
      S; (2,3)
      CLK; (2,1), (3,1)
      COMP;  (3,3)
      C;  (1,3), (2,4)
      Y;  (1,2), (3,4)
HEURISTICS:  RULE071
             RULE076
```

```
NAME:  BUS
TYPE:  SIGNAL
DESCRIPTOR:  8, BIT
CONDITIONS:  5, MA., MAX
HEURISTICS:  RULE062
             RULE065

NAME:  BIT
TYPE:  SIGNAL
MODIFIER:  TERMINAL

NAME:  EIGHT_BIT_ADDER
TYPE:  MODULE
...

NAME:  SHIFTER
TYPE:  MODULE
...

NAME:  TWOS_COMP
TYPE:  MODULE
...
```

Figure 2. Abbreviated hologram for an ALU. [4]

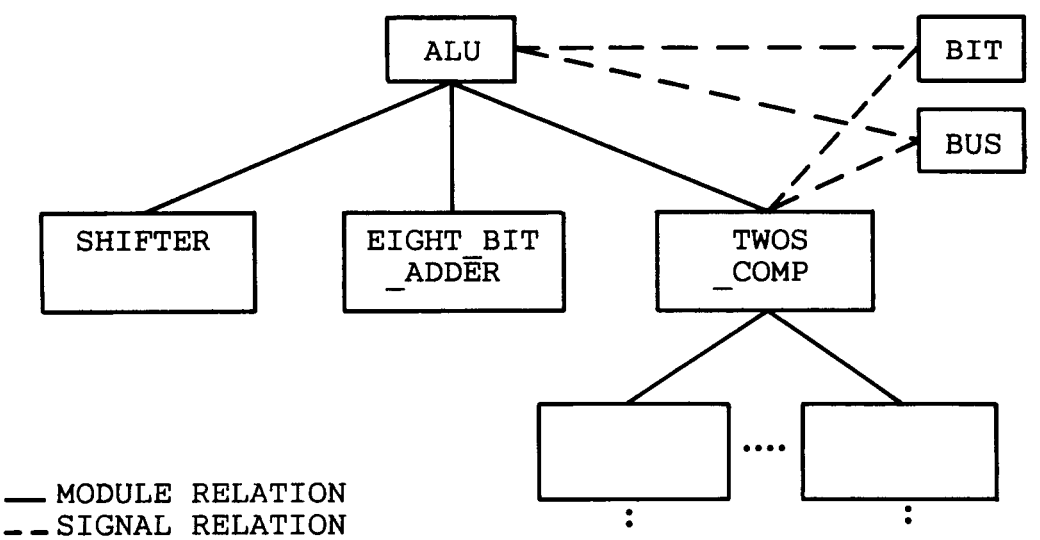

Figure 3. Abbreviated hologram structure for an ALU. [4]

semantic nets is slow since the whole network has to be traversed. But new meanings can be easily created. Pure prototype representation does not facilitate easy creation of new meanings; but comparison is easy since only the root modules and signals need to be compared. The hologram provides mechanisms for both the dynamic creation of new meanings and easy comparison.

To extend this work several designs were implemented from a hologram representation to a finished design. During these investigations, properties of the hologram required for easy automatic synthesis were developed. For automatic hardware synthesis the hologram data structure has to facilitate easy decomposition, easy similarity assessment, and should be a vehicle for name unification. (Similar modules may have different port names, and the number of ports do not have to match.) To facilitate these requirements, the hologram should contain slots of information pertaining to the detailed functioning of the hologram and attribute slots for easy determination of similarity.

3.1. Decomposition

For easy automatic decomposition and name unification the function of the hologram should be in a special form. Consider the functioning of a digital circuit. Here, the inputs are transformed to intermediate signals by a function, and these intermediate signals are transformed to the next level of intermediate signals by another function, and so on until the output signals are generated. Consider the analogy between these functions and primitives of a language. Then a language comes to mind whose primitives are the functions of actual hardware devices. Let us call this language the Actual Device Language (ADL). If the function of a hologram can be expressed in this language, the hardware design is done, in theory (on a high level), because it does not take into consideration lower level design constraints (such as propagation delay, power supply requirement etc...). Each leaf node in the design tree is an Actual Design Primitive, each intermediate node is an aggregate of Actual Design Primitives having as many subfunctions as there are submodules. Therefore, each function definition at a higher level is expressed as a number of subfunctions, where the subfunctions have subfunctions and so on.

In the decomposition or analysis phase, holograms are compared to library holograms, and if no matching (or similar) hologram is found, the hologram is simply decomposed to its subholograms. Figure 4 shows selected Actual Device Primitives, and Figure 5 depicts a hologram function which can be decomposed into subfunctions. As can be seen from the examples, the Actual Device Primitives are procedure-like and readable. Actual signal names take the place of ENABLE or INPUT1.

3.2. Similarity Assesment

The data model for the knowledge base is a hyper-semantic data model. An overview of traditional and semantic data models can be found in [5]. To compare holograms to library holograms each hologram contains slots for attributes. The collection of

a., buffer

```
INPUT --> OUTPUT
```

b., two-to-one multiplexer

```
IF      (ENABLE AND CLOCK)     THEN INPUT1 --> STORE --> OUTPUT
ELSE IF (NOT ENABLE AND CLOCK) THEN INPUT2 --> STORE --> OUTPUT
ELSE                                           STORE --> OUTPUT
```

c., D-latch with 3-state output

```
IF NOT OUT_CONT THEN IF INP_EN THEN INPUT --> STORE --> OUTPUT
                                ELSE       STORE --> OUTPUT
                     ELSE                    Z    --> OUTPUT
```

Figure 4. Example Actual Device Primitives

```
FUNCTION: MULTIPLEXER WITH 3-STATE BUFFER
FUNCTION 1.
     IF      (ENABLE AND CLK)     THEN INPUT1 --> STORE --> OUTPUT1
     ELSE IF (NOT ENABLE AND CLK) THEN INPUT2 --> STORE --> OUTPUT1
     ELSE                                        STORE --> OUTPUT1
FUNCTION 2.
     IF NOT OUTPUT2_CONTROL THEN OUTPUT1 --> OUTPUT2
                            ELSE    Z    --> OUTPUT2
```

Figure 5. Example function of a simple module

these attribute values uniquely describe the hologram. Each hologram has a function attribute (key attribute) which will place it into a device class (adder, ALU, processor...). The rest of the attributes are divided into two classes: one, the primary attributes which are associated with the functioning of the device, and second, the secondary attributes express the physical properties of the device. For example, for an adder the primary attributes would be: number of bits, full, output type..., and secondary attributes: propagation delay, fanin, fanout, power supply needed etc.

Each entity class has a restricted attribute set associated with it. This set is large enough that devices can be unique. Also each attribute of this set has a restricted domain which is the set of values an attribute can contain. Typing of the attributes is enforced. The attributes belong to the enumerated data type, and their value is their ordinal value. For each class of entities each attribute is associated with a set of heuristics, from this set a designated one will be used at the evaluation of the similarity of the attribute. The knowledge/data schema for the knowledge base is shown in Figure 6. Because the knowledge base is very large, the uniform handling of knowledge and data is necessary to ensure flexibility of design.

At the comparison of holograms belonging to the same class,

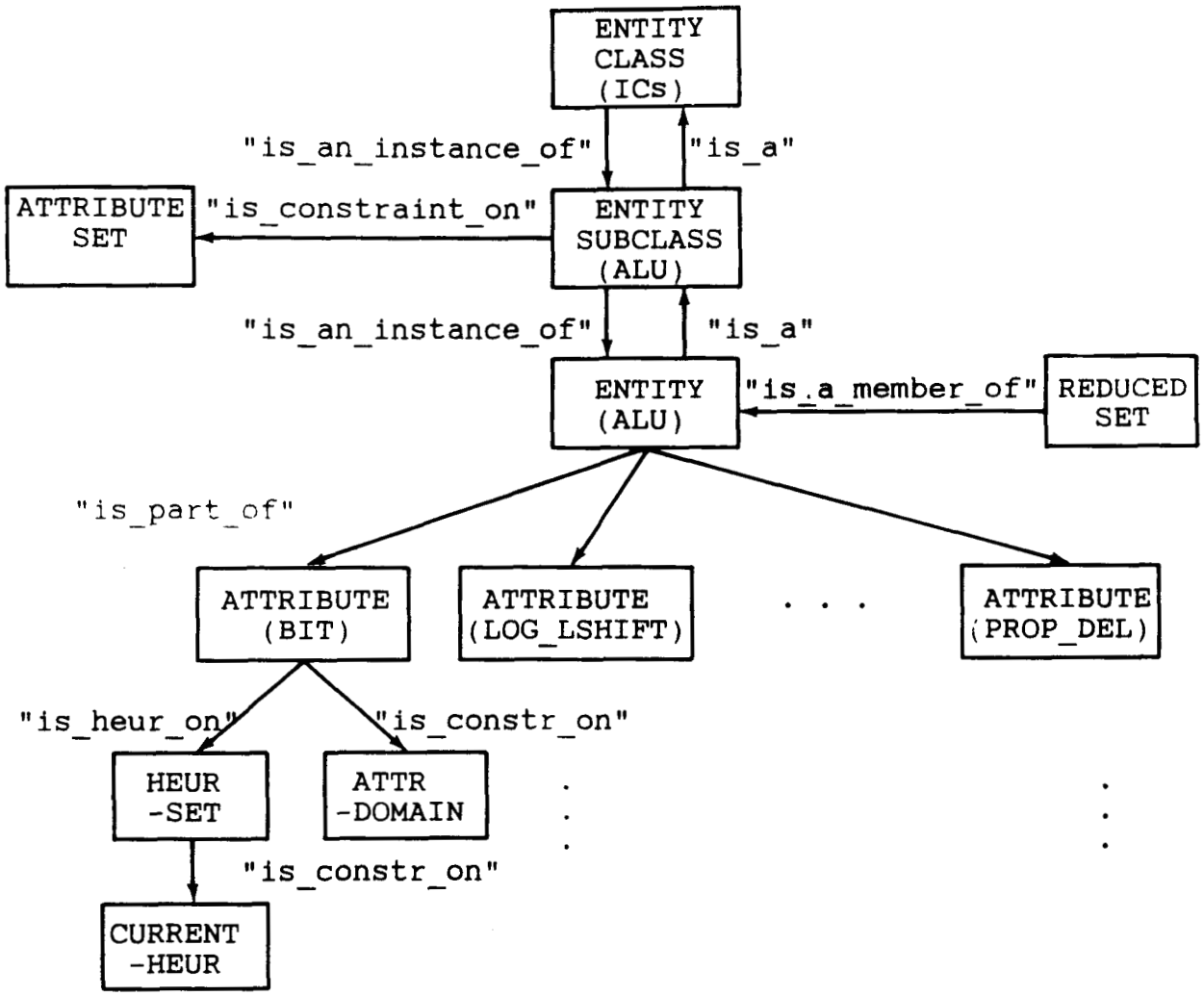

Figure 6. Abbreviated knowledge/data schema

first the primary attributes are processed. Based on the value of the attribute of the candidate device and on the value of the same type of attribute of the device to be matched, and on the associated heuristic, a score is generated. The scores of the primary attributes are added together and a set of devices with the highest score are selected for further processing. The sum of scores for the secondary attributes are generated for this set and added to the sum of primary scores. The device with the highest composite score is selected, and its similarity to the device to be matched is generated comparing its score to the maximum attainable score. Selecting actual devices using attributes and scores is shown in Table 1. Figure 7 depicts a new hologram, which contains a function definition and attributes.

3.3. Name unification

For port name unification there needs to be a way to identify which names correspond to each other in two holograms. Simple positional correspondance can be used in port lists (if there is

ATTR TO MATCH	ADDERS / PRI ATTRIBUTES	80		82		83,283		183	
		ATTR	SCORE	ATTR	SCORE	ATTR	SCORE	ATTR	SCORE
FULL	FULL	FULL	1	FULL	1	FULL	1	FULL	1
16	BIT NUMBER	1	0.06	2	0.12	4	0,25	1	0.06
-	NO OF COMP	SINGL		SINGL		SINGL		DUAL	0.06
-	EXTRA FUNC	GATED	0.8	-	1	-	1	CYSAV	0.8
	SUM		1.86		2.12		2.25		1.92

Now devices 83 and 283 are in the set.

ATTR TO MATCH	ADDERS / PRI ATTRIBUTES	SN74LS83A		SN7483A		SN74LS283		SN74S283	
		ATTR	SCORE	ATTR	SCORE	ATTR	SCORE	ATTR	SCORE
LS	SERIES	LS	1	-	0.9	LS	1	S	0.8
1	FANIN	1	1	1	1	1	1	1	1
10	FANOUT	5	0.9	5	0.9	5	0.9	5	0.9
1.5	COST	1.6	0.9	1.7	0.8	1.4	1	1.6	0.9
	SUM		6.05		5.85		6.15		5.85

Table 1. Example of selecting devices

```
HOLOGRAM DESCRIPTION: ARITHMETIC LOGIC UNIT
TYPE:  MODULE
KEY ATTRIBUTE: ALU
BIT ATTRIBUTE : 8
SERIES ATTRIBUTE: LS
FUNCTION:
   NUMBER OF SUBFUNCTIONS: 3
   FUNCTION 1:
      FUNCTION : LEFT_SHIFT_ZERO_FILL
      ATTRIBUTES : POSITIVE_EN,POSITIVE_EDGE_TRIG,3_STATE
      IF (S AND CLK) THEN LSZF(A) --> C
   FUNCTION 2:
      FUNCTION : TWOS_COMP
      ATTRIBUTES : POSITIVE_EN,POSITIVE_EDGE_TRIG
      IF(COMP AND CLK) THEN TCOMP(B) --> Y
   FUNCTION 3:
      FUNCTION : ADD
      ATTRIBUTES : POSITIVE_EN,3_STATE
      IF COMP THEN ADD(A,Y) --> C
IN:  A, BUS
     B, BUS
     S, BIT
     CLK, BIT
     COMP, BIT
...
```

Figure 7. A new abbreviated hologram

a convention for port ordering), if the number of ports and their functions match. There is a harder problem if the number of ports and/or their functions do not match. Since each hologram has a function description generated from Actual Device Primitives, the two functions can be compared and names falling in the same position textually can be unified.

4. A METHOD TO DEVELOP A ROOT LEVEL HOLOGRAM

A method to translate a VHDL code to a hologram representation without function definition and attributes has been developed by Klon [4]. To generate the function description of the hologram and the attribute set, an extension of the compiler is necessary. This extension of the compiler will recognize blocks of codes behaving as registers, adders, gates etc. An alternate method is to have device templates as standard components in VHDL which can be filled in at specification definition time. In both cases the attributes will be generated after the hologram function is defined. A method to develop chip level models in VHDL is described in [1].

5. ANALYSIS AND SYNTHESIS PHASE OF THE DESIGN

In the analysis phase the root hologram or specification hologram is decomposed into a tree of modules which have additional connections between them (i.e. signals). The modules correspond to the hierarchy of structural building blocks and the signals are the connections between them.

When a matching hologram is found in the library, it means that the hologram's score attribute is equal to the highest attainable score. Then the hologram and the subtree attached to it are copied with some name unifications. When a similar hologram is found, then the attribute deficiencies are analyzed, and an apropriate action is taken to correct the deficiency. For example if a 16 bit D-latch is needed but the hologram selected only has 4 bits, then four subholograms will be created one for each device. If there is no way to correct the deficiencies, the hologram will be decomposed to its subfunctions and if there are no subfunctions the help of the designer is requested.

Synthesis begins when the root hologram is decomposed, and all leaf nodes are actual devices. Physical device parameters are propagated upward to intermediate holograms until at the root node they are compared to the design requirements. If the design meets the requirements partially, then the source of the problem is, pinpointed and additional requirements are input to the system so as to explore alternative designs.

Once a design is finished , several alternatives can be generated by changing the VHDL code or by adding additional restrictions. To see how a VHDL description can influence the implementation, consider the following. In VHDL several styles of hardware design are supported. These styles are the structural, data-flow, and behavioral styles of descriptions. Additionally in the data-flow design style, it is possible to have control logic and data inextricably woven together; or

separate, much like in a state-machine model. These descriptions are equivalent in contextual meaning or semantics. However they imply a different hardware implementation.

6. CONCLUSIONS

In this paper a way to design digital hardware automatically is described. First a VHDL code is developed which is translated to a data structure named hologram. This data structure is a combination of production rules, frames, and semantic networks. Rules facilitate the inclusion of knowledge particular to the hologram, frames help organize heterogenous data, and the semantic networks facilitate the dynamic creation of new designs. The hologram also contains attributes for easy comparison with knowledge base holograms. The decomposition and name unification of holograms is made possible by the function description of the hologram, which is written in a language whose primitives are the functions of actual hardware devices. This hologram is input to the analyzer and synthesizer of the expert system, which after several iterations produces a hardware design. If this design only partially meets the requirements, additional requirements are given and the process is repeated until a design which meets the requirements is generated.

REFERENCES

[1] Armstrong J. R. "Chip Level Modeling with VHDL," E.E. Dept., Va., Tech, Blacksburg, Va., June 1987.

[2] Goering, R. "Silicon Compilers Bridge the Gap between Concepts and Silicon, "Computer Design", November 1987, pp. 67-80.

[3] Green C. R."Development of an Expert Hardware Synthesis System, doctoral dissertation, University of Alabama in Huntsville, Al. December 1985.

[4] Klon, P.F. "On Interfacing HDL to Knowledge Bases", doctoral dissertation, University of Alabama in Huntsville, Alabama, May 1986.

[5] Potter, W. D. and Trueblood, R. P. "Traditional, Semantic, and Hyper-Semantic Approaches to Data Modeling" Computer, June 1988, pp. 53-63.

[6] Shiva, S. G. "Automatic Hardware Synthesis", Proceedings of the IEEE, Vol. 71, No. 1, January 1983, pp.76-78.

[7] Vanderheiden, Robert M. "VHSIC: Midterm Report on a Dynamic Circuit Program, "Defense Electronics, February 1982, pp. 54-62.

[8] "VHDL Language Reference Manual", Draft Standard 1076/B, CAD Language Systems, Inc., Rockville, MD, May 1987.

[9] "VHDL Tutorial for IEEE Standard 1076 VHDL", CAD Language Systems, Inc., Rockville, MD, May 1987.